활 동 보 고 서

회차	3회차	지도교수 확인란	
일시	2019.08.07.(수) 14:00~17:00		
장소	충남대 도서관 그룹스터디룸		
주제	파이썬과 루비(2) - 함수 그리고 모듈, 객체지향 프로그래밍을 중심으로.		

학습 내용 요약

<함수와 모듈>

함수의 기본 개념은 입력값이 있으면 출력 값을 주는 코드. 이 입력값의 개수는 0개부터 여러개가 될 수도 있다. 이때까지 사용하던 print도 함수의 종류고 print나 len등은 언어에서 제공해주는 내장함수이며, 별도의 설치나 조작이 없어도 사용이 가능함.

입력값의 경우 함수 이름 뒤에 괄호 안에 넣어주면 됨. 함수 내부에서는 이렇게 전달된 입력값을 변수처럼 마음대로 사용이 가능하다. 내부에서 지역 변수로 선언해 준 것.

다만 숫자형이나 문자형 등은 아무리 함수 내부에서 값을 바꾸어도 외부에 값을 전달해준 변수는 변하지 않는다며 그것은 인자값의 복제가 일어나기 때문.

모듈은 프로그래밍에서 사용하는 부품을 말한다. 어떻게 보면 함수도 부품이라고 할수 있는데, 함수보다 더 큰 규모의 파일이 모듈이라고 할 수 있다.

이전 시간에 학습했던(2주차) '수와 계산' 파트에서 모듈에 대해 배웠던 것을 복습
= import math에서 import는 모듈을 호출하는 명령어.

math 모듈에는 ceil(올림), floor(내림), sqrt(제곱) 등의 함수가 들어있고 루비에서는 Math의 경우는, 별도의 import 없이 바로 Math를 호출해 주면 된다.

내장 모듈을 사용하면 코드를 작성하지 않고 손쉽게 문제를 해결할 수 있으며, 이러한 모듈이 없다면 직접 모든 함수를 짜야하거나, 코드가 엄청나게 길어질수 있다.

또한 코드를 한 파일에 넣어서 엄청나게 길어질 경우 변수나 함수의 이름이 충돌될 가능성이 있다.

즉, 일정한 기준에 따라 성격이 비슷한 파일들을 하나의 디렉토리에 묶어주는게 바로 모듈이며.
코드의 복잡도를 낮추고 문제를 방지하기 위한 중요한 기능을 한다.

<객체지향 프로그래밍>

모든 데이터를 하나의 물체처럼 취급해서 프로그래밍 하는 방법.
1모든 데이터를 오브젝트(object:물체)로 취급하며, 이 오브젝트에는 클래스(class:類)의 개념이 있어서 상위와 하위의 관계가 있다. 클래스의 구체적인 예가 인스턴스(instance)이다. 오브젝트 사이는 메시지의 송신으로 상호 통신한다. 가장 특징적인 것은 각 클래스에 그 메시지를 처리하기 위한 방식이 있다는 것이다. 어떤 인스턴스에 메시지가 도래하면 그 상위 클래스가 그것을 처리한다.
객체지향프로그램은 C, Pascal, BASIC 등과 같은 절차형 언어(procedure-oriented programming)가 크고 복잡한 프로그램을 구축하기 어렵다는 문제점을 해결하기 위해 탄생되었다.

절차형 언어에서는 코드 전체를 여러 개의 기능부분 즉, 인쇄하는 기능부분과 유저로부터의 입력을 받는 기능부분 등으로 분할하는데, 이와 같이 각 기능부분을 구성하는 코드를 모듈이라고 한다.
절차형 언어에서는 프로그램을 여러 기능으로 나누고 이들 모쥴을 편성하여 프로그램을 작성할 경우, 각 모쥴이 처리하는 데이터에 대해서는 전혀 고려하지 않는다. 다시 말하면 데이터 취급이 완전하지 않고 현실 세계의 문제를 프로그램으로서 표현하는 것이 곤란하기 때문에 탄생된 객체지향 프로그래밍은 객체라는 작은 단위로서 모든 처리를 기술하는 프로그래밍 방법이다. 모든 처리는 객체에 대한 요구의 형태로 표현되며, 요구를 받은 객체는 자기 자신 내에 기술되어 있는 처리를 실행한다. 이 방법으로 프로그램을 작성할 경우 프로그램이 단순화되고, 생산성과 신뢰성이 높은 시스템을 구축할 수 있다.

<팀원별 소감>

예슬: 전부터 객체지향 프로그래밍 이라는 말을 많이 접했고, 아무리 찾아봐도 제대로 이해가 되지 않았는데 이번 수업시간에 객체지향 프로그래밍에 대해 천천히 배우면서 더 자제하게 알 수 있었고, 비로소 이해할 수 있게 되었다

미규: 컨테이너와 반복문, 코드, 함수, 모듈에 대해 배웠다. 특히 지금까지는 소스 코드를 작성하는 방법에 대해만 살펴봤는데 이번 시간에는 소스 코드가 무엇인지, 컴퓨터가 내부적으로 작동하는 방법, 소스코드를 더 작성하기 위해선 어떤 원칙을 배워야 하는지에 대해 배웠다. 독일의 수학자인 라이프니츠는 '기호로 간단히 표현하는 것은 사물의 본질을 찌를 때이고, 그럴수록 생각하는 수고는 놀랄 만큼 줄어든다.'라고 말했는데 과거에 했던 말들이 현대를 살고있는 내가 배우는 코딩과도 연관이 있다고 생각하니 기분이 이상했다. 회차가 진행될수록 배우는 내용들이 점점 어려워 지는 것 같아서 스터디가 끝나고 나서도 혼자 여러번 복습해보고 어려운 부분은 멤버들과 공유해야겠다.

정란: 독학하며 강의로만 들었던 내용이 처음에는 잘 와 닿지 않았는데 회원들과 발표, 토론의 시간을 가지면서 서로 이해가 어려웠던 점을 물어보고 개인의 지식을 보충할 수 있게 되었다. 특히 함수와 모듈은 다른 프로그래밍 언어보다 실생활과 더 밀접한 관련이 있다는 것을 깨달았으며 그것들이 쓰이는 기계들을 먼저 생각하고 이론에 대입하면 공부하기가 더 수월했다.

예린: 컴퓨터 활용능력 자격증 시험을 위해서 스프레드 시트의 함수(개발자 도구에서 사용자 정의 함수를 적는다든지 프로시저를 작성할 때), 를 열심히 외운 적이 있다. 이해도 잘 안 되는 채로 열심히 외우기만 해서인지 결과는 좋지 않았다. 물론 스프레드 시트에서의 함수와 파이썬이나 루비 내에서의 함수는 다르지만, 코딩을 먼저 배우고 스프레드시트에 접근했다면 조금 더 쉬웠을 거란 생각이 강의를 듣는 내내 들었고, 코딩이 코딩 그 자체뿐만 아니라 말 그대로 '컴퓨터 활용능력'자체에도 도움이 될 거란 생각이 들었다. 또한 '모듈'이라는 개념도 새로 접했는데, 중복을 방지하고 이름이 같은 함수 또한 공존할 수 있게 설정하는 것이란 것도 알게 되었다. 코드를 입력할 땐 어렵고 오류가 나면 화가 나고 답답하지만, 그래도 새로운 것을 알아갈 때마다 세상을 넓혀가는 기분이 든다.

현지: 학습 영상을 볼 때는 다 이해되는 것 같은데 막상 내가 코딩을 하려고 하면 막막한 기분이 든다. 특히 모듈은 새로운 개념이라 이해가 어려웠지만, 동아리 회원들에게 가르쳐 준다는 느낌으로 설명을 해보니 이해가 조금 더 쉽게 되는 것 같았다. 오늘 배운 내용은 특히 배울 때는 신기하고 흥미로운 내용이기는 하지만 복습을 하지 않으면 안 될 것 같다.

학습 방법 및 과정

오늘은 각자 강의를 공부 해온 이후 함수와 모듈에 대해 설명해 보는 시간을 가졌다. 리더인 신예슬 학우부터 학번 순으로 발표하기로 정한 뒤 발표는 시작됐다. 신예슬 학우는 중복의 제거와 재사용성, 코드의 효율성에 기반하여 프로그래밍은 발전해 왔다는 말을 흥미롭게 던지며 그 중에서도 함수는 재사용성을 높이고 효율적인 코드를 만들기 위해 꼭 필요한 존재라고 답했다. 다음으로 강미규 학우가 함수의 본질적인 개념을 설명했다.

그는 함수의 기본 개념이 입력값이 있으면 출력 값을 주는 코드라며 물론 이 입력값의 개수는 0개부터 여러개가 될 수도 있다고 말했다. 덧붙여 우리가 이때까지 사용하던 print도 함수의 종류고 print나 len등은 언어에서 제공해주는 내장함수이며, 별도의 설치나 조작이 없어도 사용이 가능하다는 흥미로운 사실로 주위를 환기 시켰다.

거기에 이정란 학우가 사용자 지정 함수를 만들 수도 있다는 말을 꺼내며 함수의 예시를 노트북 창에 띄웠다.

위의 코드가 지정 함수에 대한 예제인데, 그렇게 효율적인 코드는 아니다. def로 함수를 정의 하고, 함수명 뒤 괄호에 인자를 넣고 내부에는 함수의 동작 코드를 명시한다. 유의 할 점은 루비에서는 코드 블록의 마지막을 end로 감싸줘야 한다는 것이다. 이렇게 만들어진 함수를 사용하기 위해서는 함수이름 뒤에 괄호와 인자를 넣어주면 된다. 만들어진 func 함수를 실행하면 'Hi!'가 출력된다. 예시

```python
# python
def func():
    print('Hi!')
func()

# ruby
def func()
    puts('Hi!')
end
func()
```

의 함수는 입력값과 출력값이 없다. 그럼 출력값을 만들어 주기위해서 어떻게 해야할까?라는 질문에 김예린 학우가 설명했다. '예시의 함수에 출력값을 만들어 주기 위해서는 함수 내부에서 모든 코드를 다 실행하고 출력할 값을 반환해 주면 된다.'

```python
# python
def func():
    return 'Hi!'
print(func())

# ruby
def func()
    return 'Hi!'
end
puts(func())
```

이러한 반환은 return이라는 키워드로 이루어지며, 함수를 호출한 곳으로 값이 전달된다. 따라서 위의 코드에서는 함수를 호출한 곳으로 'Hi!'문자열이 반환되며, 이를 출력 했을 때 정상적으로 문자열이 나오게 된다는 사실을 김예린 학우가 깔끔하게 설명했다.

마지막으로 입력값에 대해서는 김현지 학우가 발표했다.

```python
# python
def Func(num):
    return 'Func'*num
print(Func(3))

# ruby
def Func(num)
    return 'Func'*num
end
puts(Func(3))
```

입력값의 경우 앞서 언급했듯이, 함수 이름 뒤에 괄호 안에 넣어주면 된다. 함수 내부에서는 이렇게 전달된 입력값을 변수처럼 마음대로 사용이 가능하다. 내부에서 지역 변수로 선언해 준것이라고 생각

하면 된다.

김현지 학우는 주의해야 할 점으로 숫자형이나 문자형 등은 아무리 함수 내부에서 값을 바꾸어도 외부에 값을 전달해준 변수는 변하지 않는다며 그것은 인자값의 복제가 일어나기 때문이라는 설명으로 마무리 하였다.

다음으로 모듈에 대해 발표하는 시간을 가졌다. 이번에는 아까와 반대된 순서대로 발표했다.

모듈이라고 하는 것은 프로그래밍에서 사용하는 부품이다. 어떻게 보면 함수도 부품이라고 할수 있는데, 함수보다 더 큰 규모의 파일이 모듈이 된다.
이에 대해 김현지학우가 아래와 같이 쉽게 설명해주었다.

"예를 들어, 학교에서 학생들을 가르치다가 학생들이 늘어남에 따라 학년으로 분리한다고 쳐요. 그러다가 학년이 엄청 많아져서 규모가 커지면 반이라는 단위로 나눠게 되잖아요.
만약 학년과 반의 개념이 없고 학교만 있다면, 학생들이 무질서하게 분포하고 있으니까 문제가 생기게 되겠죠?

학년은 교육 정도나 나이에 따라 분류되고, 반은 생일이나 키, 이름에 기반한 번호로 줄세워지게 될거고. 이와 같이 일정한 기준에 따라 성격이 비슷한 파일들을 하나의 디렉토리에 묶어주는게 바로 모듈이에요.

즉 모듈은 코드의 복잡도를 낮추고 문제를 방지하기 위한 중요한 기능이에요."

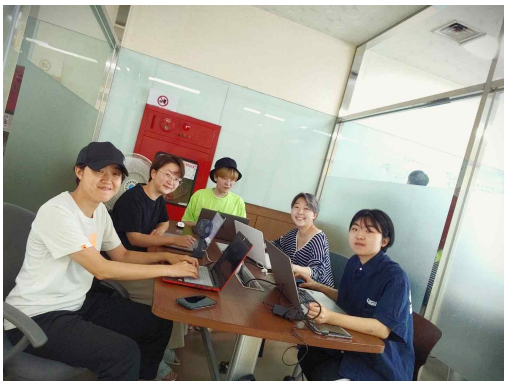
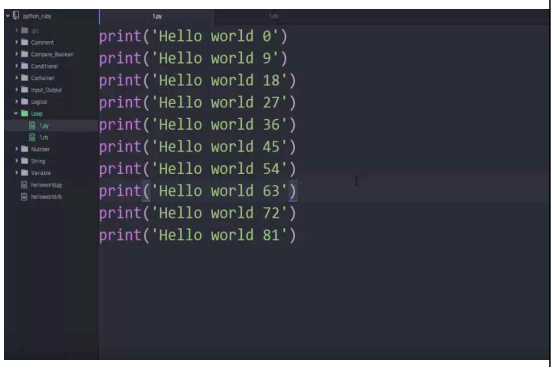
다음으로 김예린 학우가 사실 우리는 모듈에 대한 것을 사용해 보았다고 했다. 그것이 바로 전 시간에 배웠던 '수와 계산' 파트에서, import math를 사용했는데, import는 모듈을 호출하는 명령어라는 것이다.

이정란 학우가 그에 덧붙여 설명했다. 그 math 모듈에는 ceil(올림), floor(내림), sqrt(제곱) 등의 함수가 들어있고 루비에서는 Math의 경우는, 별도의 import 없이 바로 Math를 호출해 주면 된다는 내용이었다.

따라서 이런 내장 모듈을 사용하면 코드를 작성하지 않고 손쉽게 문제를 해결할 수 있다는 내용을 강미규 학우가 보충 설명을 해주었다. 그는 추가로 이러한 모듈이 없다면 직접 모든 함수를 짜야하거나, 코드가 엄청나게 길어질수 있음을 의미한다는 말도 덧붙였다.

그리고 코드를 한 파일에 넣어서 엄청나게 길어질 경우 변수나 함수의 이름이 충돌될 가능성이 있다는 신예슬 학우의 말을 끝으로 우리의 수업 내용 정리 및 발표는 마무리 되었다.

활동 사진 첨부

<table>
<tr>
<td>학습활동①</td>
<td> 
스터디룸에서 찍은 단체 인증샷</td>
<td>학습결과물①</td>
<td> 

'반복문' 관련 강미규 학우의 실습화면 크롭</td>
</tr>
<tr>
<td>학습활동②</td>
<td>두산백과
객체지향프로그래밍
[object-oriented programming]

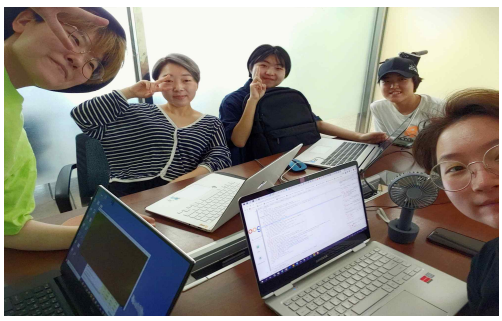
요약 모든 데이터를 오브젝트(object:물체)로 취급하여 프로그래밍 하는 방법으로, 처리 요구를 받은 객체가 자기 자신의 안에 있는 내용을 가지고 처리하는 방식이다.

이 개념은 1960년 중엽에 유행한 시뮬레이션 언어의 SIMURA에서 유래한 것이다. 모든 데이터를 오브젝트(object:물체)로 취급하며, 이 오브젝트에는 클래스(class:類)의 개념이 있어서 상위(上位)와 하위(下位)의 관계가 있다. 클래스의 구체적인 예가 인스턴스(instance)이다. 오브젝트 사이는 메시지의 송신(送信)으로 상호 통신한다. 가장 특징적인 것은 각 클래스에 그 메시지를 처리하기 위한 방식이 있다는 것이다. 어떤 인스턴스에 메시지가 도래하면 그 상위 클래스가 그것을 처리한다. 현재 오브젝트지향개념은 프레임 표현형식과 융합하여 인공지능을 위한 소프트웨어 기법(技法)의 하나로 되어 있다.

객체지향프로그램은 C, Pascal, BASIC 등과 같은 절차형 언어(procedure-oriented programming)가 크고 복잡한 프로그램을 구축하기 어렵다는 문제점을 해결하기 위해 탄생된 것이다. 절차형 언어에서는 코드 전체를 여러 개의 기능부분 즉, 인쇄하는 기능부분과 유저로부터의 입력을 받는 기능부분 등으로 분할하는데, 이와 같이 각 기능부분을 구성하는 코드를 모듈이라고 한다. 절차형 언어에서는 프로그램을 여러 기능으로 나누고 이들 모듈을 편성하여 프로그램을 작성할 경우, 각 모듈이 처리하는 데이터에 대해서는 전혀 고려하지 않는다. 다시 말하면 데이터 취급이 완전하지 않고 현실 세계의 문제를 프로그램으로서 표현하는 것이 곤란하다.

이러한 절차형 프로그래밍이 가지는 문제를 해결하기 위해 탄생된 객체지향프로그래밍은 객체라는 작은 단위로서 모든 처리를 기술하는 프로그래밍 방법으로서, 모든 처리는 객체에 대한 요구의 형태로 표현되며, 요구를 받은 객체는 자기 자신 내에 기술되어 있는 처리를 실행한다. 이 방법으로 프로그램을 작성할 경우 프로그램이 단순화되고, 생산성과 신뢰성이 높은 시스템을 구축할 수 있다.

'객체지향 프로그래밍'읽기 자료

 
인증샷2</td>
<td>학습결과물②</td>
<td>

```
class Cal
  def initialize(v1,v2)
    @v1 = v1
    @v2 = v2
  end
  def add()
    return @v1+@v2
  end
  def subtract()
    return @v1-@v2
  end
end
c1 = Cal.new(10,10)
p c1.add()
p c1.subtract()
c2 = Cal.new(30,20)
p c2.add()
p c2.subtract()

```

이정란 학우의 실습화면 크롭</td>
</tr>
</table>